A Software Defined Radio Based on the Goertzel Algorithm for the DCF77 Longwave Time Signal

1st Leander Hackmann
Master Embedded Systems Engineering
FH Dortmund
Dortmund, Germany
Mat.-Nr.: 7217912, leander.hackmann001@stud.fh-dortmund.de

***Abstract*—The DCF77, a long wave time signal transmitter that is located near Frankfurt am Main in Germany, is the primary source of time information for radio-controlled clocks in Europe. Its signal consists of a phase and an amplitude-modulated part, both containing the time information. Devices that use the signal usually consist of an amplitude modulation long wave receiver, often realized by using a dedicated integrated circuit and an application processor. In this paper, an alternative approach for the demodulation of the amplitude-modulated part of the signal, without the need for specialized analog cicruitry and with only a minimal amount of simple hardware components, is presented. This is done by utilizing direct-sampling software-defined radio techniques together with leveraging the efficiency of the Goertzel algorithm. As a result, the presented approach removes the need for most of the usually employed analog circuitry, leaving only a front-end amplifier, and therefore effectively proposes a new type of design where the receiver is implemented inside the application processor. The employed lightweight algorithms keep it realizable on a system with limited resources, which is shown in an example implementation on a Arm Cortex-M3 microcontroller.**

***Index Terms*—DCF77, Goertzel, Digital Signal Processing, Time Signal, Software Defined Radio**

I. Introduction

Radio controlled clocks are very popular because they do not require periodical re-adjustment in comparison to common mechanical or quartz-powered clocks. Instead, the information about the current time is obtained from a RF-signal, transmitted by an official authority that ensures its availability and correctness. The probably most important time signal in Europe is transmitted by the station DCF77 from Mainflingen, a city in Germany that is close to Frankfurt am Main. The station is operated by a private company that is commissioned by the Physikalisch-Technische Bundesanstalt (PTB), which also supplies the current time information, generated by two caesium clocks and two caesium fountains. The time signal's operating frequency lies in the long wave area at 77.5 kHz, simple modulation techniques (amplitude and phase-modulation) are used and it is transmitted with 100 kW Effective Isotropic Radiated Power (EIRP) to enable the reception of the signal in relatively close distances (up to 2000 km) with simple technical equipment [2].

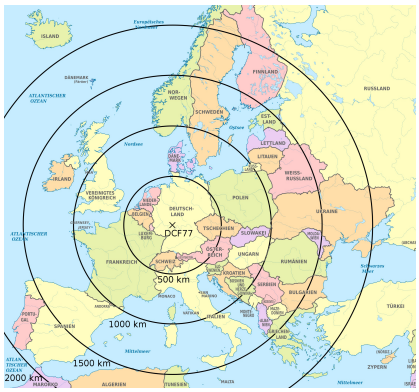

Fig. 1. Coverage of the DCF77 signal. Adapted from [1]

End-user devices usually make use of the amplitude-modulated part of the signal by using an amplitude modulation receiver, often a by employing specialized integrated circuits, together with a rather big ferrite-rod antenna to obtain the time code signal. This time code signal then needs further hardware, often in form of a microcontroller used as the applicaton processor, to decode and retrieve the actual time information. With the increasing availability of computation power and Digital Signal Processing (DSP) capabilities inside cheap mixed-signal microcontrollers, which are already used to process the time information, the idea to replace the analog receiver portion with a Software Defined Radio (SDR) is more or less obvious. Moving the receiver into the application processor would allow for a simpler hardware design by shrinking down the complexity and the parts count of the external analog circuitry. In fact, only a tuned antenna with a basic pre-amplifier is needed.

In this paper, a straightfoward principle for designing a lightweight receiver for the DCF77, based on SDR-techniques, is presented. After taking a look at the structure of the DCF77 signal, the theoretical operation principle of the receiver is explained and shown in a MATLAB simulation. To further underline the simplicity and the possibility to implement the principle on a system with limited resources, a demonstration on a Arm Cortex-M3 microcontroller is shown.

II. The DCF77 signal

The DCF77 station operates in the long wave area at a frequency of $77.5\,\mathrm{kHz}$. Its operation started with transmitting a frequency standard in 1959. In 1973, the transmission of the current time information was added to the signal by using amplitude modulation. To allow a reception under more challenging conditions and with a higher resolution and accuracy, the same time information is also added to the signal since 1983 by using phase-modulation [3]. The two different parts of the signal that result out of the parallel amplitude and the phase-modulation can be used for different use-cases that are separated by the requirements regarding the resolution, accuracy, and the reliability of the obtained time information. Whereas the amplitude-modulated part is mainly used for general-purpose time application with low resolution requirements, the phase-modulated part is usually used for specialized applications that require a higher resolution, accuracy, and reliability. Although being considered as low in comparison to what is achievable by analyzing the phase-modulated part, the usually observed accuracy uncertainity when using the amplitude-modulated part it is still well below $1\,\mathrm{s}$, being $100\,\mathrm{ms}$ [2]. When using the phase-modulated part, a standard deviation of $\pm 2\,\mu\mathrm{s}$ to $\pm 22\,\mu\mathrm{s}$ from the time of the atomic clock to the time of the receving device can be observed [4].

As the amplitude-modulated part allows simplistic receiver designs through its straightfoward modulation scheme, it is used in the principle presented in this paper and therefore a focus is put on explaining this part of the DCF77 signal. The following explainations in this paper therefore refer explicitly to the amplitude-modulated part. The time information, represented as a series of binary symbols, is modulated onto the carrier with a frequency of $77.5\,\mathrm{kHz}$ by using Amplitude-Shift Keying (ASK). Every symbol has the duration of exactly $1\,\mathrm{s}$, with a part that is either $100\,\mathrm{ms}$ or $200\,\mathrm{ms}$ long where it has a value equal to 0. This the duration of this "gap" is used to encode respectively a 0 or a 1 and effectively switches the strength of the carrier between $15\,\%$ and $100\,\%$ [5].

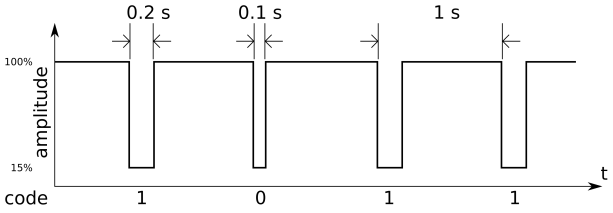

Fig. 2. Symbol encoding of the DCF77 signal. From [1]

To allow the detection of a new minute, the gap is left out between the last (59) second of the ongoing and the first (0) second of the next minute. By using this scheme, a dataframe consisting out $59\,\mathrm{Bit}$, starting at second 1, can be transmitted every minute. Bit 0 to 19 are used to send encrypted weather data, to notify the PTB in case of errors in the transmitter, to announce an upcoming switch to or from Daylight Saving Time (DST), to show if DST is currently enabled or not, and to announce an upcoming leap second [5]. As the decryption of the weather date requires a specialized third-party integrated circuit and the notification bit is only important for the PTB, these parts of the frame are discarded by most applications. The leftover Bits 20 to 58 are used to represent the current time and date in Binary Coded Decimal (BCD) format, including parity bits for recognizing errors.

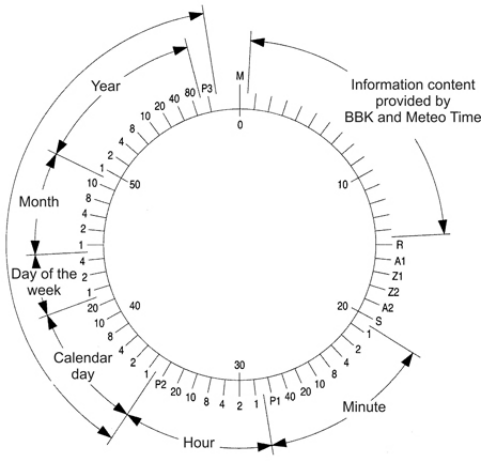

Fig. 3. Data frame encoding of the DCF77 signal. From [5]

This means that a receiver needs to run at least for $39\,\mathrm{s}$ after powerup to receive one whole block containing the current time information. In reality, this period is usually longer because the receiver needs to adapt to the current receiving conditions. These conditions underly a lot of influences, including temporary fading, electromagnetic noise from the close environment or varying signal strengths during different times of the day.

III. Principle of the receiver

As seen in the description of the DCF77 signal, retrieving back the binary time signal from the amplitude-modulated part is more or less trivial. Conventional applications execute this task by using analog receiving circuits, usually by using a simple amplitude modulation receiver together with a tuned ferrite-rod antenna, acting as a Bandpass Filter (BPF), to filter out the irrelevant part of the spectrum. Moving the receiver into the digital domain allows more advanced techniques to retrieve this binary time signal, rather than demodulating the signal with a diode detector inside the amplitude modulation receiver. Because the information is contained only in the amplitude of the signal, and therefore inside the signal strength at the receiver, it can also be obtained by doing a continuous analysis of the spectrum at the carrier's frequency of $77.5\,\mathrm{kHz}$. This task will be carried out by using the Goertzel algorithm, which is presented later. The structure of the whole receiver then simplifies to only two external hardware parts (besides the application processor) and four internal software parts.

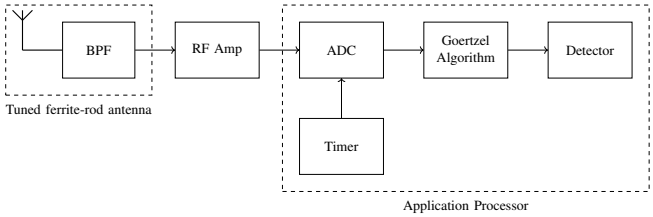

Fig. 4. Block diagram of the receiver

The tuned ferrite-rod antenna is kept to filter out the irrelevant spectral components, leaving only the DCF77's component at 77.5 kHz. This signal is then amplified by a RF amplifier and sampled at a certain sampling rate f_s using the Analog to Digital Converter (ADC) of the application processsor. A timer producing a frequency equal to f_s is used to trigger the sampling. The samples are passed to the Goertzel algorithm to analyze the amplitude of the DCF77 signal. The result of the analyzation is then passed to a detector to reconstruct the binary time signal.

Before discussing the simulation of the whole reciever, the internal parts executing the sampling, the spectral analysis and the detection are explained in more detail, as they use some special techniques that should be introduced and explained beforehand.

A. Bandpass sampling the DCF77 signal

Lowpass sampling is probably the most commonly used sampling technique. Stated by the Nyquist criterion, it requires the sampling rate f_s to be at greater or equal to twice of the frequency of the highest frequency f_h component in the signal to be sampled:

$$f_s \geq 2 \cdot f_h \tag{1}$$

Only when this condition is satisfied, the original signal can be reconstructed from the samples and it must be ensured that the signal contains no spectral elements that might violate the condition [6]. This is usually done by filtering the signal through a lowpass before sampling.

Besides lowpass sampling, a signal can also be bandpass sampled if it is has a limited bandwidth B, defined by the frequency of the lowest spectral component f_l and the highest spectral component f_h:

$$B = f_h - f_l \tag{2}$$

The center frequency f_c is then located in the middle between f_l and f_h:

$$f_c = \frac{f_l + f_h}{2} = f_l + \frac{B}{2} \tag{3}$$

Signals that fulfill these requirements can also be sampled with a lower f_s and still be reconstructed, if special care is taken when selecting the actual value of f_s. The ranges of possible values for f_s are defined by the conditions

$$\frac{2 \cdot f_c - B}{m} \geq f_s \geq \frac{2 \cdot f_c + B}{m+1} \tag{4}$$

and

$$f_s \geq 2 \cdot B \tag{5}$$

with m being an arbitrary, positive integer [6]. During this sampling process, the signal is also shifted in frequency. This can be imagined by a sum of convolutions of the initial spectrum with dirac delta functions, located at positive and negative integer multiples of f_s [7].

As the DCF77's signal fulfills the stated requirements after it is received by the tuned ferrite-rod antenna and therefore bandpass filtered, it qualifies for being bandpass sampled. Its center frequency f_c lies at 77.5 kHz with a theoretical maximum bandwidth B of 2.4 kHz [2]. The effective bandwidth is much smaller in reality but 2.4 kHz will be used for finding a proper sampling frequency to provide enough safety margin. With (4), a table containing possible ranges for f_s can be created.

TABLE I
POSSIBLE RANGES FOR f_s WITH $f_c = 77.5\,\text{kHz}$ AND $B = 2.4\,\text{kHz}$

m	Lower boundary of f_s $\left\lceil \frac{2 \cdot f_c + B}{m+1} \right\rceil$	Upper boundary of f_s $\left\lfloor \frac{2 \cdot f_c - B}{m} \right\rfloor$
1	78 700 Hz	152 600 Hz
3	52 467 Hz	76 300 Hz
4	39 350 Hz	50 866 Hz
5	31 480 Hz	38 150 Hz
6	26 234 Hz	30 520 Hz
7	22 486 Hz	25 433 Hz
8	19 675 Hz	21 800 Hz
⋮	⋮	⋮
31	4919 Hz	4922 Hz
32	4770 Hz	4768 Hz
33	4630 Hz	4624 Hz

It is visible that with increasing values for m, the value of f_s and also the distance between the lower and the upper boundaries are decreasing. At $m = 32$, the condition of (5) is violated and at $m = 32$ the condition of (4), making $m = 31$ the last valid range. Choosing a sampling rate can now be done with this table.

A lower sampling rate effectively shrinks the requirements regarding computational power, memory size and the ADC's speed, but simultaneously decreases the distance between the lower and the upper limits of the sampling rate. This then leads to higher requirements regarding the adherence to the exact sampling rate. A choice providing a good tradeoff between these effects $f_s = 24\,\text{kHz}$ at $m = 7$, because it offers a distance of 2647 Hz between the upper and lower limits with the sampling frequency lying almost in the middle [8]. The DCF77 signal is then, as visible in Fig. 5, downconverted and centered at

$$f_c^{'} = f_c - 3 \cdot f_s = 77.5\,\text{kHz} - 3 \cdot 24\,\text{kHz} = 5.5\,\text{kHz} \tag{6}$$

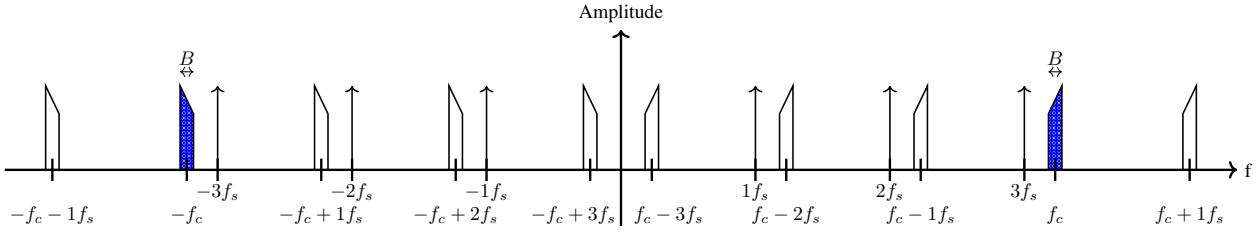

Fig. 5. Spectrum after bandpass sampling the DCF77 signal at $f_s = 24\,\text{kHz}$. The original spectral parts are filled with blue dots.

caused by the convolution of the initial spectrum with the dirac delta impulse located at $-3 \cdot f_s$. The following block, the Goertzel algorithm, therefore needs to analyze the spectral component at $f_c^{'} = 5.5\,\text{kHz}$.

B. The Goertzel Algorithm

The standard approach to analyze a spectrum in the digital domain after sampling is the well-known Discrete Fourier Transform (DFT), implemented by using the Fast Fourier Transform (FFT). This yields an analysis of the complete spectrum, producing $\frac{N}{2} + 1$ relevant results for equally distributed frequencies ("frequency bins") within a range from $0\,\text{Hz}$ to $\frac{f_s}{2}$ where N is the number of samples and f_s is the sampling frequency [6]. The signal strength, and therefore the binary time signal of the amplitude-modulated part, can then be examined by observing the magnitude of the curresponding frequency bin over time. Because of only one frequency bin being relevant, it is obvious that a full FFT produces a lot of overhead and redundant calculation operations in this use-case. This overhead can be avoided by using the Goertzel algorithm instead of the FFT, which is able to calculate the value of one specific frequency bin with a smaller amount of needed calculations. It was invented in 1958 by Gerald Goertzel and since then is most commonly used in the detection of tones employed in Dual-tone Multi-frequency (DTMF) signaling [9]. The working principle of the algorithm can be percieved as a second-order Infinite Impulse Response (IIR) filter (see (7)) where the results $y[n-1]$ and $y[n-2]$ of the output $y[n]$ are stored in the states Q_1 and Q_2, so they are still available after the execution of the filter [10].

$$y[n] = x[n] + 2 \cdot cos(\omega_g) \cdot y[n-1] - y[n-2] \tag{7}$$

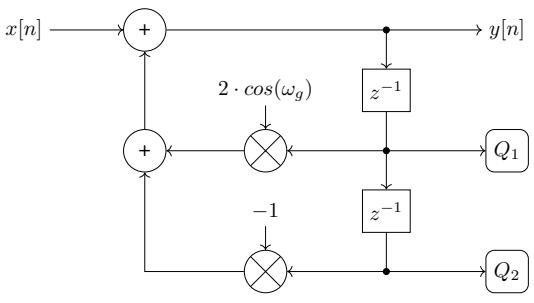

Fig. 6. Block diagram of the Goertzel algorithm without the final calculation of the squared magnitude

The system takes the input sequence $x[n]$ with length N and is executed exactly N times. After the execution, the squared magnitude of f_a, the frequency to be analyzed inside $x[n]$, can be calculated using the values of the two states Q_1 and Q_2:

$$|X(f_a)|^2 = Q_1^2 + Q_2^2 - Q_1 \cdot Q_2 \cdot 2 \cdot \cos(\omega_g) \tag{8}$$

The constant $2 \cdot cos(\omega_g)$, often called the "Goertzel coefficient", is calculated before the execution and used to adapt the algorithm to the frequency to be analyzed. The Value of ω_g itself is calculated based on a integer number k and N. The integer number k is a rounded fraction based on N, the frequency to be analyzed f_a, and the the sampling frequency f_s:

$$k = \left\lfloor \frac{N \cdot f_a}{f_s} \right\rceil \tag{9}$$

$$w_g = \frac{2\pi \cdot k}{N} \tag{10}$$

Just as with the conventional FFT, the amount of needed calculations for executing the Goertzel algorithm is based on the number of samples. In total, $N + 1$ multiplications and $2 \cdot N + 2$ additions are needed, which can be seen from the formulas for the IIR-part and the final calculation of the squared magnitude. For example, a length of processing $N = 512$ samples requires in 513 multiplications and 1026 additions to calculate the squared magnitude of the desired frequency. In comparison, the typical radix-2 FFT, needs $\frac{N}{2} \cdot \log_2(N)$ complex multiplications and $N \cdot \log_2(N)$ complex additions to be completed. With $N = 512$ this would result in 2034 complex multiplications and 4608 complex additions.

With this simple example, it is already visible that the Goertzel algorithm can reduce the amount of needed resources compared to a full FFT, when only one or a small amount of frequency bins should be analyzed. It is therefore used to observe the DCF77 signal's amplitude over time by being executed periodically. As the gap of the encoded symbols is eiter $100\,\text{ms}$ or $200\,\text{ms}$ long, the duration of the analyzation is set to $10\,\text{ms}$ to provide a good tradeoff between resolution in time and a longest possible analyzation time to leverage the DFT gain. The square roots of the results are taken to finally calculate the magnitude of the DCF77 signal's amplitude and then passed to a detector to obtain the binary time signal. Because of the duration of the analyzation being equal to $10\,\text{ms}$, the sampling rate after the Goertzel algorithm is equal to:

$$f_{s,g} = \frac{1}{10\,\text{ms}} = 100\,\text{Hz} \tag{11}$$

C. The detector

The output of the Goertzel algorithm is already following the waveform of the binary time signal but still includes fluctuations due to noise and fading. To reconstruct the original waveform of the binary time signal, it is processed inside a detector. As the values of the Goertzel algorithm will follow the pattern of the signal's amplitude being switched between $15\,\%$ and $100\,\%$, the detection can be done by comparing the value of the current sample to a threshold. To compensate the changing of magnitude of the Goertzel algorithm's values that is caused by the current receiving situation, an Exponential Moving Average (EMA) is calculated by using (12) and employed to create an adaptive threshold [11].

$$y[t] = \alpha \cdot x[t] + (1 - \alpha) \cdot y[t-1] \tag{12}$$

This part can also be represented as a block diagram:

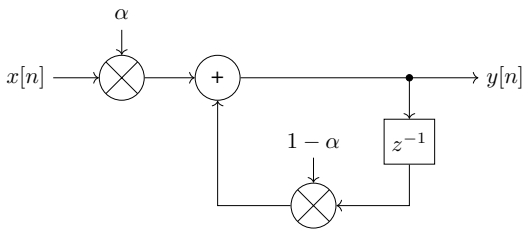

Fig. 7. Block diagram showing the calculation of the EMA

The value of the smoothing factor α was calculated to take the samples of the last $10\,\mathrm{s}$ into account:

$$\begin{aligned} \alpha &= \frac{2}{N+1} = \frac{2}{10\,\mathrm{s} \cdot f_{s,g} + 1} \\ &= \frac{2}{10\,\mathrm{s} \cdot 100\,\mathrm{Hz} + 1} \approx 0.001998 \end{aligned} \tag{13}$$

This provides a stable and slowly changing average, as it is putting more weight on the past samples and leaves minimal influence to the most recent sample. The decision is then done by comparing the current sample to a threshold calculated by multiplying the current value of the average with 0.575 (the middle bewteen $15\,\%$ and $100\,\%$) and a scaling factor of $\frac{1}{0.8725}$:

$$threshold = 0.575 \cdot \frac{1}{0.8725} \cdot y[t] \tag{14}$$

The scaling factor is needed as the average percentage of time where the signal where is equal to $100\,\mathrm{percent}$ is $87.25\,\%$ (assuming an equal distribution between symbols representing a binary 1 and a 0). If the value of the current sample is above this threshold, it is treated as a 1. Otherwise it is treated as a 0. The series of ones and zeros is then following the waveform shown in Fig. 2 and can be used to decode the current time.

IV. Simulation of the receiver in MATLAB

To verify and further explain the working principle of the receiver, a simulation of the whole system was created using MATLAB. The whole simulation was divided into five sections to allow for a clear structure:

1) Simulation Parameter Section
2) Signal Creation Section
3) Goertzel Algorithm Section
4) Detection Section
5) FFT and Plot Section

These sections will be discussed in the following.

A. Simulation Parameter Section

In the Simulation Parameter Section, static parameters for the following sections of the simulation are defined. These include the sampling frequency f_s, the duration of the whole simulation, the SNR that the signal should have after passing the Additive White Gaussian Noise (AWGN)-Channel, the frequency to be analyzed by the Goertzel algoritm, the duration of the Goertzel algorithm segments, and the time that should be taken into account by the EMA. By adjusting these parameters, the receiving situation and the receiver's behavior can be altered to, e.g. to simulate different receiving conditions and examine the system's performance.

B. Signal Creation Section

The Signal Creation Section implements the creation of the amplitude-modulated DCF77 signal. First, a time vector with the length of the simulation's duration and a step size that depends on f_s is created and its length stored in a variable. A vector containing the carrier signal at $f_c = 77.5\,\mathrm{kHz}$ by using this time vector is created afterwards. For the amplitude modulation of the carrier, the time code signal containing the binary encoded symbols needs to be created. As this signal is used for multiplication with the carrier, it needs to consist of rects with a period length of $1\,\mathrm{s}$ and a gap of either $200\,\mathrm{ms}$ or $100\,\mathrm{ms}$ at the beginning. These rects are prepared by creating and filling two vectors with the matching amount of zeros followed and filling the rest with ones. A vector containing the time code signal with alternating symbols is created by concatenating these vectors inside a for loop. The number of iterations is given by rounding up the simulation's duration, because each symbol has the duration of $1\,\mathrm{s}$. The vector is then cut down afterwards to exactly match the length of the time vector and therefore the duration of the simulation. To finally create the DCF77 signal, the vector with the time code signal is scaled and multiplied element-wise with the vector containing the carrier signal. This signal is then passed through a AWGN-Channel, creating a signal with a defined SNR to allow the simulation of different receiving conditions.

C. Goertzel Algorithm Section

As described in the chapter explaining the receiver's working principle, the sampled DCF77 signal is processed in segments whose size is defined by the duration of the Goertzel algorithm's execution. Before processing the samples, the parameters needed for the segment-wise execution are prepared. These include the segment size, the number of segments, and a time vector containing the exact points in time where segments' results were available (later used for plotting). The parameters for the Goertzel algorithm itself, k,

ω_g and $2 \cdot cos(\omega_g)$ are then calculated by using the formulas (9) and (10).

Three variables, Q_0, Q_1, and Q_2, are initialized for the implementation of the Goertzel algorithm's formula (see (7) and (8)) and vector for storing the segments' results is prepared. The segment-wise analyzation is then implemented by using an outer and an inner for-loop. The outer for-loop iterates over the number of segments, first resetting Q_0, Q_1, and Q_2 followed by calculating the index of the first and the last sample of the current segment to provide the limits for the inner for-loop. Inside the inner for-loop, the formula (7) of the IIR-Part is implemented by calculating the new output of the structure and shifting the previous results in each iteration. After all samples of the segment were processed (the inner for-loop finished), the magnitude is calculated by the implementation of (8), using the values stored in Q_1 and Q_2 and taking the square root of the result. The result is then stored inside the prepared vector for the segments' results.

D. Detection Section

The Detection Section processes the Goertzel algorithm's results of the segments to retrieve back the time code signal. At the beginning, three vectors are initialized to store the value of the EMA, the detector's threshold, and the time code signal. The parameter α of the EMA is calculated by the implementation of (13) and stored in a variable. The segments are then processed inside a for-loop. First, the EMA is calculated by the implementation of (12) and stored inside a vector. Then, the threshold is calculated based on the value of the EMA and the scaling factor (see (14)) and stored inside a vector. The decision whether the result of the current section was caused by the signal's amplitude being equal to either $15\,\%$ or $100\,\%$ is done by comparing the value of the result to the calculated threshold. If the value is smaller than the threshold, it is assumed that it was caused by the transmitter's amplitude being equal to $15\,\%$ and a 0 is stored in the vector of the reconstructed time code signal. If the value is bigger than the threshold, it is assumed that it was caused by the transmitter's amplitude being equal to $100\,\%$ and a 1 is stored in the vector of the reconstructed time code signal.

E. FFT and Plot Section

To allow the a visualization and verification of the simulation's results, a section for plotting was added. This also includes two FFTs, analyzing signal before and after passing the AWGN-Channel to allow the obervation of the SNR. The FFTs are prepared before the plotting takes place by using MATLAB's internal functions for calculating and shifting the results. The scaling is adjusted to normalize the DFT gain and the logarithm of the results is calculated. Besides the calculation of the FFTs themselves, a vector containing the bins' frequencies is prepared for plotting. The results of the two FFTs are then plotted in figure.

In a second figure, multiple time domain plots are arranged to allow the observation of the receiver's stages. These include plots of the time code signal, the DCF77 signal with and without noise, the results of the Goertzel algorithm's segments together with the threshold of the detector, the value of the EMA, and the reconstructed time-code signal.

F. Results

Multiple simulations with different values for the SNR starting from $5\,\mathrm{dB}$ to $-7\,\mathrm{dB}$ were executed to verify and evaluate the receiver's capabilities to reconstruct the DCF77's time code signal in different receiving conditions. From these simulations, the most relevant results occured in the case where the SNR was equal to $-5\,\mathrm{dB}$, because this was the lowest SNR where the reciver was still able to retrieve back the time code signal without errors. The frequency domain plots of the two FFTs, shown in Fig. 8, verify the DCF77 signal's aliases to show up exactly at $\pm 5.5\,\mathrm{kHz}$. From the plot showing the DCF77 signal after passing the AWGN-channel, it can be seen that the noise floor is located at approximately $-50\,\mathrm{dB}$ while the peaks of the DCF77 signal have a value of approximately $-8\,\mathrm{dB}$. The difference of $42\,\mathrm{dB}$ should not be confused with the SNR, as the power of the noise added by the AWGN-Channel is spread evenly over all frequencies. The correct SNR was verified to be exactly $-5\,\mathrm{dB}$ by using MATLAB's built-in command "snr()" to calculate the SNR by comparing the before and after adding noise.

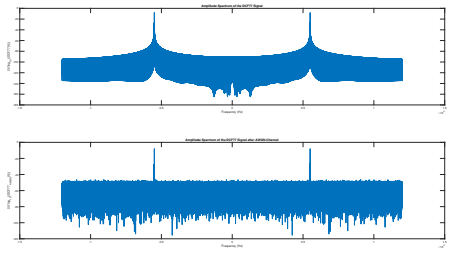

Fig. 8. Frequency domain plots of the simulation run with a SNR of $-5\,\mathrm{dB}$

The time domain plots of Fig. 9 show the creation of the DCF77 signal and the internal operation of the receiver. The first plot shows the time code signal that should be retrieved back by the receiver to allow a visual verification that the DCF77 signal is modulated retrieved back correctly. It is visible that each symbol has the expected length of $1\,\mathrm{s}$ with a gap of either $200\,\mathrm{ms}$ or $100\,\mathrm{ms}$ at the beginning. From the second plot, the amplitude can be seen switching between 0.15 and 1, representing the changes between $15\,\%$ and $100\,\%$. The third plot shows the DCF77 signal with added noise and an average amplitude of approximately 3.6 with the orignal waveform not being visible anymore. The fourth plot shows the results of the Goertzel algorithm segments together wit the value of the calculated threshold. The operation of the Goertzel algorithm can be verified, as the calculated magnitude of the frequency bin is clearly following the waveform of the time code signal. The value propagation of the threshold is as slow as expected with the value only experiencing minor drops during the gaps of the signal. A value where the detector starts to take valid decisions is reached after approximately

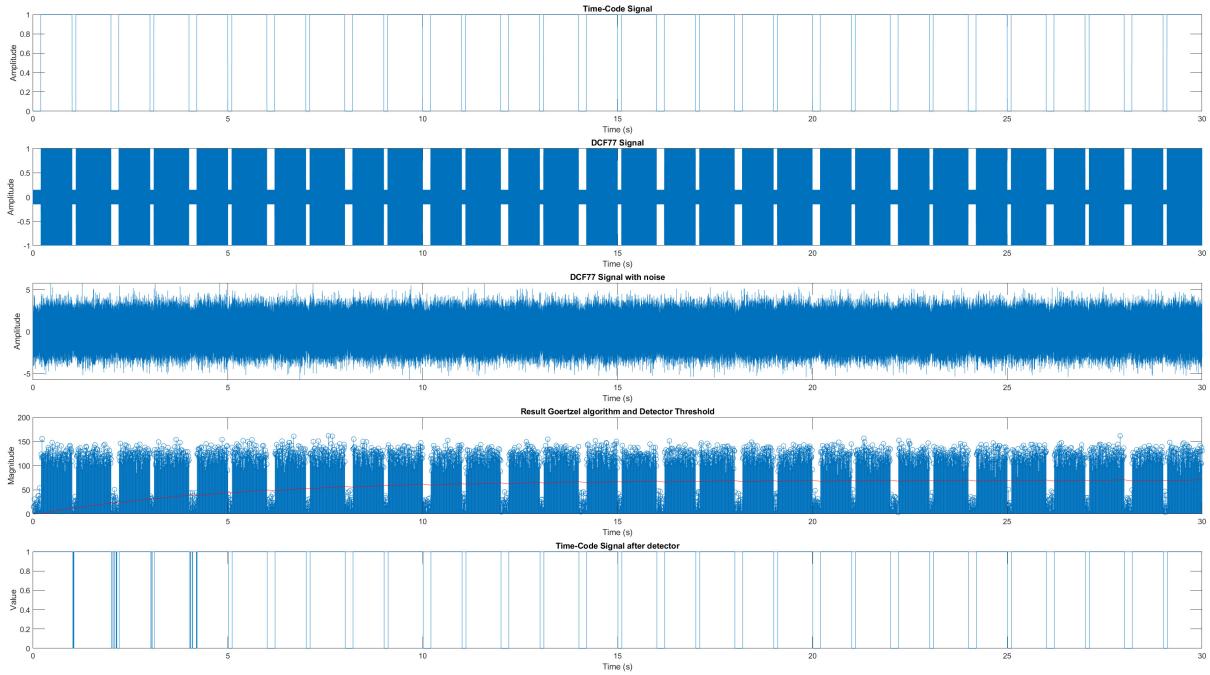

Fig. 9. Time domain plots of the simulation run with a SNR of $-5\,\mathrm{dB}$

$5\,\mathrm{s}$. The threshold then approaches asymptotically the value where it is exactly located in the middle between the area of values representing an apmlitude of $15\,\%$ and the area of values representing $100\,\%$. As the value of the EMA is proportional to the threshold, the plot is left out. In the fifth plot, the retrieved time code signal is visualized. It is obvious that the signal contains errors until the threshold has reached a level where valid decisions are taken by the detector. After $5\,\mathrm{s}$, when the threshold reached this value, no more errors occured.

All simulation runs with a smaller SNR contained an increasing amount of errors, caused by the noise to raise or drop single points above or below the threshold, placing them in the area where they were detected to correspond to the opposite symbol. This effect can be imagined by thinking of the height of the area where the results are distributed by the noise is increased until both areas touch the threshold and start to overlap. Single points ending up in the other area are then interpreted as the wrong symbol.

V. Demo implementation

The receiver's structure, shown in Fig. 4, was implemented on a STM32F103C8T6, an Arm Cortex-M3 microcontroller, to provide a brief proof of concept. For an optical verification, the obtained time code signal is used to turn on and off a Light-Emitting Diode (LED), but this signal could also be used to decode the DCF77's transmitted time data. As a tuned antenna together with a pre-amplifier is needed to filter out the DCF77 signal and amplify it to a level where it can be measured by the microcontroller's internal ADC, a minimalistic circuit consisting of a tank circuit tuned to $77.5\,\mathrm{kHz}$ and a two-stage transistor amplifier was built based on the circuit shown in [12]. The rest of the of the receiver's structure was implemented in software using C and single-precision floating-point for the caluclations.

A timer was set up to create an internal Pulse Width Modulation (PWM) signal to trigger the conversion of the microcontroller's internal ADC at the sampling frequency of $24\,\mathrm{kHz}$. The samples are transferred using circular Direct Memory Access (DMA) into a buffer of the size of 480. This size was chosen because it is equal to twice the amount of samples that are needed for the analyzation of one segment. During runtime, the DMA hardware module fills this buffer and two interrupts are executed when the buffer was filled half way and when it is completely full. Because of the circular setup of the DMA, it fills the buffer continuously and wraps around back to the first address, after reaching the buffer's end. This allows a continuous operation of the application with a precise sampling frequency while eliminating the need to handle ADC conversions manually. As a segment is ready to be analyzed with every interrupt, the operations needed to process the signal processing are executed inside this Interrupt Service Routine (ISR). The result of each segment after the detector is then used to turn on or off a General Purpose Input Output (GPIO) that is connected to a LED.

Although its rather simplistic implementation, the demo implementation worked right away. The behavior of the receiver

that was observed in the simulation, first outputting a series of wrong results when adapting to the current receiving conditions, is also noticable when running the demo application. After powerup or changes in the environment and therefore during the stabilization of the threshold to its new value, the LED shows arrythmic behavior for approximately $5\,\mathrm{s}$ before the application continues to operate correctly. Other than that, the demo implementation shows that the presented principle also works in reality and that it can be implemented with low effort. Based on this structure, the code can be expanded to a full clock by decoding the information from the retrieved time code signal. The code is available in the paper's GitHub repository [13].

VI. Conclusion

After the theoretical discussion, the simulation, and the proof of concept by showing a demo implementation, it can be seen that the presented principle can be used to realize a receiver for the DCF77 time signal with minimal effort regarding hardware and signal processing. This is achieved by moving most of the signal processing into the digital domain of the application processor to leverage the effects of bandpass sampling, the efficiency of the Goertzel algorithm, and the simplicity of a detector based on an EMA. In addition, the proposed architecture also shrinks down the need for external specialized analog circuitry, leaving only a tuned antenna and a pre-amplifier. While the the complexity that is typically associated with analog receivers is reduced, the principle also increased adaptability to changing receiving conditions. As all algorithms were designed to require a minimum of resources, the principle can be applied to enable systems with restricted resources to make use of the DCF77 time signal.

Acknowledgment

This work was inspired by a project that was published by an anonymous author in a german forum about electronics (see [12]). The autor of this paper, L. Hackmann, wants to thank for the knowledge and ideas that were provided through freely accessible entries.

References

[1] -, "DCF77," 2024, Wikipedia entry. [Online]. Available: https://de.wikipedia.org/wiki/DCF77

[2] D. Priester and P. Hetzel and A. Bauch, "Zeit- und Normalfrequenzverbreitung mit DCF77," PTB Mitteilungen, 2004, [Online]. Available: https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/4.4_zeit_und_frequenz/4.42/dcf77.pdf

[3] D. Priester and P. Hetzel and A. Bauch, "Zeit- und Normalfrequenzverbreitung mit DCF77: 1959-2009 und darüber hinaus," PTB Mitteilungen, 2009, [Online]. Available: https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/4.4_zeit_und_frequenz/pdf/2009_Bauch_PTBM__DCF77.pdf

[4] P. Hetzel, "Time dissemination via the LF transmitter DCF77 using a pseudo-random phase-shift keying of the carrier," PTB Braunschweig, 1988, [Online]. Available: https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/4.4_zeit_und_frequenz/pdf/5_1988_Hetzel_-_Proc_EFTF_88.pdf

[5] -, "DCF77 webpage," 2024, Official page of the PTB [Online]. Available: https://www.ptb.de/cms/en/ptb/fachabteilungen/abt4/fb-44/ag-442/dissemination-of-legal-time/dcf77.html

[6] R. G. Lyons, "Understanding Digital Signal Processing," Third Edition, Prentice Hall, 2010.

[7] J. Hoffmann, "Sampling bandpass signals," 2021, Internet Article. [Online]. Available: https://www.dsprelated.com/showarticle/1407.php

[8] F. Zaplata and M. Kasal, "SDR Implementation for DCF77,", 2013, IEEE Paper. [Online]. Available: https://ieeexplore.ieee.org/document/6530943

[9] -, "Goertzel algorithm," 2024, Wikipedia entry. [Online]. Available: https://en.wikipedia.org/wiki/Goertzel_algorithm

[10] K. Banks, "The Goertzel Algorithm," 2002, Internet Article. [Online]. Available: https://courses.cs.washington.edu/courses/cse466/12au/calendar/Goertzel-EETimes.pdf

[11] R. G. Lyons, "Setting the 3-dB Cutoff Frequency of an Exponential Averager," 2012, Internet Article. [Online]. Available: https://www.dsprelated.com/showarticle/182.php

[12] -, "ATTINY85 als DCF77-Empfänger," 2021, Forum Entry. [Online]. Available: https://www.mikrocontroller.net/topic/529477

[13] L. Hackmann, "SiCo1_Project," 2024, GitHub Repository. [Online]. Available: https://github.com/lhckmn/SiCo1_Project